DOI:
10.1007/s10639-016-9493-x

Document Version
Publisher's PDF, also known as Version of record

Educ Inf Technol
DOI 10.1007/s10639-016-9493-x

Computer science in K-12 school curricula of the 2lst century: Why, what and when?

Mary Webb[1] · **Niki Davis**[2] · **Tim Bell**[3] · **Yaacov J. Katz**[4] · **Nicholas Reynolds**[5] · **Dianne P. Chambers**[6] · **Maciej M. Sysło**[7]

Abstract In this paper we have examined the position and roles of Computer Science in curricula in the light of recent calls for curriculum change and we have proposed principles and issues to consider in curriculum design as well as identifying priority areas for further research. The paper is based on discussions within and beyond the International Federation of Information Processing (IFIP) Education Community since 2012 as well as an analysis of curriculum developments in five different countries. Emerging themes have been discussed with reference to important perspectives from curriculum theory including "powerful knowledge" as a key element of entitlement and management of the growth of expertise. Based on this analysis we have identified areas of consensus as well as constraints, risks and issues that are still subject to controversy. There is an emerging consensus of the importance of Computer Science and the nature of its "powerful knowledge". Furthermore current understanding of the opportunities and benefits for starting to learn Computer Science early in primary schools has

IFIP is the leading multinational, apolitical organisation in Information and Communication Technologies and Sciences; IFIP was originally set up under the auspices of UNESCO and continues to have a formal consultative status within UNESCO.

✉ Mary Webb
mary.webb@kcl.ac.uk

[1] King's College London, Franklin-Wilkins Building, Stamford Street, London SE1 8WA, UK

[2] University of Canterbury e-Learning Laboratory, Christchurch, New Zealand

[3] Department of Computer Science and Software Engineering, University of Canterbury, Christchurch, New Zealand

[4] Michlala-Jerusalem Academic College and Bar-Ilan University, Jerusalem, Israel

[5] Melbourne Graduate School of Education, The University of Melbourne, Melbourne, Australia

[6] The University of Melbourne, Melbourne, Australia

[7] UMK Toruń, University of Wrocław, Wrocław, Poland

Published online: 28 April 2016

identified this early start as an entitlement and equity issue. There is a strong consensus that teacher professional development in Computer Science Education is critical for supporting curriculum change and is currently a major challenge in many countries. Other key issues include understanding how the growth of expertise affects potential structure and sequencing in the curriculum and the balance of content. Further considerations include how new technological opportunities interact with pedagogical approaches and can provide new potential for the growth of expertise.

Keywords Computer science · Curriculum design · Entitlement · Powerful knowledge · Informatics · Primary education · Secondary education

1 Introduction

The need for education systems to respond to changes brought about by technological developments through "digital literacy" and the use of technologies to enhance opportunities for learning is generally accepted. However the importance and roles of Computer Science in the curriculum are less well understood and are the focus of this paper. Computer Science is the term used in this paper for the academic discipline underlying the general area of computing; these and other related terms are defined and discussed in the next section. Arguments for the inclusion of Computer Science (also known in German as Informatik) in the curriculum are compelling and were summarised at EDUSummIT 2015, a meeting of International experts, including policymakers, practitioners and researchers on IT and education as: economic, social and cultural (Webb et al. 2015). The economic rationale rests not only on the need for a country to produce computer scientists to sustain a competitive edge in a world driven by technology but also on the need for Computer Science-enabled professionals in all industries to support innovation and development. The social rationale emphasises the value in society of a diverse range of active creators and producers rather than just passive consumers of technology. Such capability provides people with power to lead, create and innovate within society. The cultural rationale rests on enabling people to be drivers of cultural change rather than having change imposed by technological developments.

However, because the relative importance of Computer Science, as an underlying academic discipline, has been subject to change and not been well understood (Denning 2007, 2009, 2013), Computer Science became lost or squeezed from curricula in some countries in recent years (see for example The Royal Society 2012; Wilson et al. 2010). Consequently serious concerns have been raised about this demise of Computer Science leading to calls for all students to develop key understanding, skills and thinking approaches that emerge from Computer Science before graduating from secondary schools and probably in primary school. Curriculum change has followed from these calls in many countries including the UK (The Royal Society 2012; Wilson et al. 2010; Seehorn et al. 2011), Canada (http://cacsaic.org/HowAlbertaGotCS) throughout Europe (Joint Informatics Europe and ACM Europe Working Group on Informatics Education 2013) Australia (http://www.australiancurriculum.edu.au/technologies/digital-technologies/rationale) and New Zealand (Bell et al. 2012). At the same time some other countries, such as Israel, Poland and Cyprus, have maintained their particular Computer Science curricula and continued their development since the 1980s.

This paper, written by members of a Task Force of the Education Committee (TC3) of the International Federation of Information Processing (IFIP) aims to examine the roles for Computer Science and the support for the rationales in the curriculum, outlined above, in the light of these recent developments. We build on previous discussions within IFIP TC3 conferences in Manchester 2012, the World Conference on Computers in Education 2013 in Poland and the Key Competences in Informatics and ICT (KEYCIT) Conference 2014, Germany. More specifically this position paper arises from a panel discussion at the 2015 conference in Lithuania in which perspectives from five different countries with different traditions of curricula were presented, analysed and their implications discussed with conference participants.

This paper explains the background to the discussions that have taken place in IFIP TC3 meetings and the terminology used in the paper, before reviewing developments in five different countries where recent curriculum developments have addressed some of the issues under discussion. We compare and contrast the responses in these countries and analyse the reasons for differences. In order to support this analysis we briefly review curriculum theory to identify important considerations and constraints and relate these to the emerging themes. Based on this analysis we recommend principles to consider in curriculum design and remaining issues and areas for further research.

2 Background

Discussions at IFIP Conferences since 2012 have established that within the IFIP Education Community there is strong agreement that new technologies are enabling new ways of thinking in education, new cultures of learning and that curriculum and assessment need to change to accommodate new opportunities. These discussions also highlighted that there are a range of views among professionals working in the area of Computer Science and ICT in education about the roles of Computer Science/Informatics in the curriculum.

Between 2012 and 2014 the debate was enthusiastic and suggested that reaching a clear consensus about a vision for the curriculum and how to frame the design of curricula for Computing/Informatics/digital literacy was challenging. Table 1 summarises key ideas that arose together with an estimate of the level of consensus amongst the participants (Webb 2014).

The range of views and incomplete consensus indicated in Table 3 as well as in other debates about the curriculum for Computing vs digital literacy indicate the extent of the challenge of establishing the roles of Computer Science and designing a curriculum framework. Key contentious aspects apparent from Table 1 are whether or not ***all*** students should study Computer Science and for what purpose: digital citizens, informed consumers, industry aware professionals? These debates are happening around the world as countries consider how children should be educated in these areas, and the situation is often exacerbated because of the lack of understanding of the terminology (a vicious cycle due to the subjects not being part of the schooling system!) For example, Computer Science is often equated with programming (Denning 2009), or might be considered to be entirely mathematical or based on particular technologies.

Table 1 Views that emerged from the panel discussion in Toruń, WCCE 2013 (Webb & Reynolds 2013; Webb 2014)

Key idea/question about Computing curricula	Level of consensus
Computer Science and digital literacy are complementary – both are needed in the school curriculum	High
Need room for flexibility in interpretation	High
What is the importance of Computer Science for general education?	High consensus that this question is important
Problems of defining terms	Consensus that terminology is important and difficult
We need to develop aware citizens – not necessarily creators but more than consumers	Contentious - needs further consideration
Teaching children to be aware, not necessarily how to create software or code	Contentious - needs further consideration
Current trend is a grass roots movement that appears to have joined forces and coordinated. At the heart of it is an understanding that Computing is essential for all children but also a need for opportunities for career paths and citizenship	Fairly high
A set of concepts based on Computer Science should be defined as a basis for the curriculum – some concepts have a long shelf-life whereas others are short-lived	Fairly high
Computer Science is for everyone	Contentious - needs further consideration
Towards a curriculum framework: When – from the beginning What – clear examples How – basic principles Who – concerns with initial teacher training and professional development	The key principles of what needs to be decided or agreed.

In order to define a vision or framework to inform curriculum development, we need to define what is the range and scope of the subject, the key ideas and subject matter in the field(s). At the same time it will be essential to clarify why these are important for the schooling sector, to enable movement towards a vision and set of principles for the curriculum design and perhaps a framework. Thus key questions emerging are:

1) What is the range of skills and understanding that should be developed?
2) Are such skills and understanding necessary for everyone?
3) At what age should such education commence? If so, to what extent should it be and possibly remain compulsory?
4) What pedagogical approaches are likely to be appropriate, and how do they vary with age and other factors?

In answering these questions we acknowledge that curriculum design is complex. No single theory of curriculum is commonly accepted that can provide us with a basis for developing our vision for curricular design (see for example Pacheco 2012). Therefore this paper draws upon, in particular, the concept of "powerful knowledge" as a key element of entitlement (Young 2013) and the management of the growth of expertise (Winch 2013).

The variation in terminology in relation to Computing/ICT has been a source of much confusion so we begin by defining the terms used here (see Table 2). The Royal Society Report (2012) provided some useful definitions based predominantly on the situation in the UK in 2012 and these form the basis for definitions in this paper with some further clarification as explained below. K-12 is used to clarify our focus on the primary and secondary school sectors (aged approximately 4–19), excluding schools/departments within tertiary education.

Informatics is slightly broader than Computer Science, but is the term used widely across Europe to refer to this discipline, for example the Joint Informatics Europe & ACM Europe Working Group on Informatics Education use the term Informatics to "cover the entire set of scientific concepts that make information technology possible" (2013 p. 9). Another term that arises from Computer Science and is important in many recent curriculum specifications is computational thinking (Papert 1980, 1996; Wing 2006). It would be possible to devote an entire paper to discussing how to define computational thinking (Barr and Stephenson 2011; Voogt et al. 2015) but we will adapt the definition provided by the Royal Society (see Table 2) that emphasises its relation to Computer Science but also indicates the broad relevance of computational thinking for many other areas of the curriculum and life in general. In summary, in this paper the terminology shown in Table 2, which is largely based on the Royal Society Report, will be used and when we refer to Computer Science we will assume that arguments also apply to Informatics.

The distinction between computational thinking and programming is subtle; in principle computational thinking does not require programming at all, although in practice, representing a solution to a problem as a program provides a perfect way to

Table 2 Terminology used in this paper

Information Technology (IT) – The use of computers, in industry, commerce, the arts and elsewhere, including using software packages, aspects of IT systems architecture, human factors, project management etc. (Note that this is from the Royal Society Report and is the title of courses in the UK at GCSE and A-level; it emphasises the assembly, deployment and use of digital systems and the Royal Society report notes that "this is narrower than the use in industry, which generally encompasses Computer Science as well" (p.5).)
Computer Science – "The scientific discipline encompassing principles such as algorithms, data structures, programming, systems architecture, design, problem solving, etc." (The Royal Society Report, p.17). In addition to principles and a stable set of concepts Computer Science incorporates rigorous techniques, methods and ways of thinking including "computational thinking".
Computing – the broad subject area incorporating IT, Computer Science, digital literacy and problem solving in this context deploying computational thinking. Computing is now the title for the new curriculum in the UK; in Australia and New Zealand "Digital Technologies" is the equivalent term used in curricula.
Digital literacy – "the general ability to use computers. This will be written in lower case to emphasize that it is a set of skills rather than a subject in its own right. "(The Royal Society 2012 p.5).
Informatics- the entire set of scientific concepts that make information technology possible (Joint Informatics Europe and ACM Europe Working Group on Informatics Education 2013)
Computational thinking –recognising aspects of computation in the world that surrounds us, and applying tools and techniques from Computer Science to understand, reason and solve problems in relation to both natural and artificial systems and processes (adapted from The Royal Society 2012 p. 29).
Programming – a process involving: analysis and understanding of problems; identifying and evaluating possible solutions; generating algorithms; implementing solutions in the code of a particular programming language; testing and debugging, in order to formulate solutions into executable computer programs.

evaluate the solution, as the computer will execute the instructions to the letter, forcing the student to refine their solution so that it is very precise. At the lower level of primary school, this might be as simple as programming a robotic toy (such as the "Bee-Bot") to follow forward/left/right movements to get to a desired location. Thus aspects of Computer Science, including programming, provide an ideal way of developing computational thinking which learners can then be encouraged to apply more broadly as a problem solving strategy (see for example Barr and Stephenson 2011 for discussion of the ways in which computational thinking may be applied in a range of curriculum areas).

We now examine vignettes from five different countries with a view to identifying how key issues and considerations play out in their approaches to curriculum development and to identify common themes and differences.

3 Vignettes of curriculum development initiatives

3.1 A view from the UK

Review of the ICT curriculum in the UK (The Royal Society 2012) identified a need for fundamental reform: a major concern was that the curriculum had become unbalanced with too much emphasis on basic digital skills at the expense of deeper understanding of concepts. The new curriculum in England for Computing introduced in 2014 (Department for Education 2013) recognises the value of Computer Science as the underlying academic discipline, forefronts computational thinking and expects pupils to "understand and apply the fundamental principles and concepts of Computer Science" as well as being able to "analyse problems in computational terms, and have repeated practical experience of writing computer programs in order to solve such problems". This does not mean that the "old ICT curriculum" has been dropped completely: students are still expected to be able to evaluate and apply information technology and to be "responsible, competent, confident and creative users of ICT".

Those of us who have been working in Computing Education in the UK for many years are pleased and relieved by these curriculum changes: in recent years many of the teachers, with in-depth understanding of Computer Science, have been deeply frustrated by constraints of the school curriculum. So there are now networks of teachers and computer scientists in the UK working very hard to make this new curriculum work and to support teachers in developing their knowledge and pedagogy (e.g. Computing at School: http://www.computingatschool.org.uk/ and Teaching London Computing: http://teachinglondoncomputing.org/). We recognise that there are high risks and major challenges to address, in particular: insufficient Computing teachers with the appropriate knowledge of the subject matter; persuading schools to find sufficient time for the new curriculum; ensuring assessment supports and enables exciting and challenging learning in Computer Science and identifying appropriate pedagogical approaches to achieve this.

One of the important and more general issues arising from the experience in England is how to avoid such curriculum degeneration in future. There are several important factors that contributed to the demise of Computer Science in the curriculum including problems with the assessment and accountability system which encouraged schools to

adopt easy exam subjects; weak specification of the curriculum leading to too much ambiguity; lack of understanding of most people of the importance of Computer Science. This last factor remains a significant problem: even people in the IT industry are prone to state that Computing is just a practical subject and there is no need for the underlying theory. There is also a current problem with the perception of programming as just coding rather than understanding it as a problem solving process.

The approach we have taken in England is to specify a strong and challenging curriculum content of knowledge, understanding and skills underpinned by Computer Science and emphasising computational thinking up to the age of 14. From 14 to 16 students are still expected to develop their understanding of computing but have the opportunity to specialise in Computer Science or IT as well as continuing to develop their digital literacy. Developing appropriate pedagogical approaches for students of a range of ability remains challenging. On the Teaching London Computing Project (http://teachinglondoncomputing.org/), a professional development programme for existing ICT teachers across London, we emphasised the importance of combinations of "unplugged" activities (Bell and Newton 2013) as well as practical hands-on experience. Unplugged activities do not use computers but are designed for learning important Computer Science concepts through paper-based scenarios and kinaesthetic learning experiences designed to focus on understanding the concepts without being distracted by needing to master tools and techniques such as programming.

3.2 A view from Aotearoa New Zealand

Aotearoa New Zealand, especially the region of Canterbury where a number of software companies with a global reach are based, provides an interesting perspective on Computer Science education which has recently been reintroduced into the school curriculum. Canterbury is also home to the Computer Science Education research group that has developed Computer Science Unplugged (Bell and Newton 2013), which is used world-wide, and leads international collaborations to produce strategically important research and development such as the Computer Science Field Guide (CSfieldguide.org.nz), an online textbook initially for New Zealand students that is being adapted and translated for other countries. These resources are also part of the University of Canterbury postgraduate programme for teachers of Computer Science Education.

Computer Science disappeared from the high school curriculum for a number of years and was introduced again in 2011 as an examination subject for seniors in high schools (National Certificate of Educational Achievement (NCEA) Digital Technologies standards (15–18); NCEA is the summative final examination system in New Zealand) (Bell 2014). While there has been considerable success from these changes (including large improvements in the diversity of students taking up careers in computing), it has also provided evidence that delaying the introduction of computing concepts to the last 2–3 years of school is not ideal. At this late stage the acquisition of programming knowledge and skills appears to be particularly challenging for students, and even more so for teachers, who need to introduce a completely new subject in the first year that students are facing external assessment for qualifications that are used beyond school (Thompson et al. 2013). Not only would earlier preparation reduce this pressure, but the level of engagement and diversity is most likely to benefit from exposure to the concepts before high school (Duncan et al. 2014).

New Zealand universities have recognised the importance of professional development for school teachers and their difficulties in acquiring enough knowledge and skills as mature adults (Bell et al. 2012). At Canterbury a course for practicing teachers has been developed that involves them actively developing curriculum resources and we are looking to develop more professional development opportunities, including most recently a course in programming that incorporates industry experience.

The challenge of having Computer Science and Programming introduced so late in a school career has led to a review of the curriculum, stimulated by the "Curious Minds" initiative (www.curiousminds.nz). The national "Review of the positioning and content of digital technologies in The New Zealand Curriculum and Te Marautanga o Aotearoa" is considering which level topics around computational thinking should be introduced, including the possibility of introduction in the first year of compulsory schooling; also how the curriculum can be best positioned to support teachers and learners, either as part of the existing Technology area of the national curriculum or as an area in its own right.

3.3 Digital technologies curriculum in Australia

Digital Technologies is a compulsory subject (Foundation to Year 10) in the Australian Curriculum which was launched in 2015. This curriculum is built on the notion that computing is a specialised learning area (different from ICT) that needs its own specification and a way of assessing achievement through mandated Achievement Standards; something that had not occurred in previous curricula where ICT was seen as integrated and could often be neglected. This curriculum (Australian Curriculum Assessment and Reporting Authority (ACARA) 2014) is built on five Key Concepts of Abstraction, Data collection (representing and interpreting), Specification, Digital systems, and Interactions and impacts (p. 26). The use of these five Key Concepts as the basis for the curriculum moves this curriculum from a purely Computer Science curriculum (although it does contain aspects of Computer Science including programming) to one that attempts to address Digital Technologies as a whole discipline, a discipline where the actions and interactions of humans and computers is of as much importance as the specific knowledge and skills required to think computationally. This curriculum assumes that students, even as young as five, have the capacity and the right to develop the skills and knowledge required to operate effectively and ethically in a digital world.

A pilot project, in the state of Victoria, which introduced the curriculum earlier than the rest of Australia, investigated the ways in which teachers went about implementing the new curriculum (Reynolds and Chambers 2014, 2015). Their study indicated that teachers were capable of understanding the complexities of the new curriculum with adequate support. It also showed that, depending on their perceptions of their own knowledge and capabilities, teachers adopted three different broad approaches to curriculum implementation: 1) applying existing practices to new curriculum; 2) applying new curriculum to existing practices or 3) new practices for a new curriculum. The project showed that teachers can build appropriate learning contexts for young children and for older students in relation to the new Digital Technologies curriculum when given focused support for developing their subject and pedagogical knowledge.

3.4 A view from Israel

Computer Science is a major subject offered in a small but significant number of Israeli high schools that are located at the upper end of the high school technology track. These are usually elite institutions where very talented students study Computer Science, mathematics, physics, chemistry and biology (biotechnology) (following Barak et al. 2000). The Computer Science curriculum in these elite schools relies on core aspects present in the both the mathematics and the physics curricula, thus a significant number of weekly hours in both mathematics and physics are mandatory for students who study Computer Science as a stand-alone subject. Therefore students who study Computer Science need to be able to cope with the demands and pressures of the advanced high school mathematics and physics curricula.

However, the mainstream of high school students in public high schools do not study Computer Science per se as a stand-alone subject but are instructed in digital and computer literacy as a major medium and methodology that contributes to their learning in all the subjects that they study (following Cheema and Zhang 2013). An intrinsic facet of the promotion of digital and computer literacy per se is the teaching of basic computational thinking in order to enhance the mastery of problem solving and analytic thinking of high school students, bearing in mind the fact that these higher order thought processes pervade all subjects studied in the school curriculum. The Ministry of Education in Israel has a well-organised and detailed strategy that directs high schools in both the enhancement of Computer Science as a stand-alone high school subject as well as the promotion of digital and computer literacy as a must for any successful high school student in all subject areas under study (Naim 2010). The rationale behind the centrality of digital and computer literacy in Israeli schools is based on the findings of Alsafran and Brown (2012) who concluded that computer use in education is one of the most potent and significant means and platforms for instruction and learning in the 21st century. Thus it has become imperative that, in their pre-service education, Israeli teachers develop knowledge and computer skills that allow them to undertake teaching based on the transferral of knowledge, as well as on higher order thinking processes, within their sphere of expertise while utilising available digital databases. Thus teachers need to be less dependent on their own knowledge of subject matter, acquired during their pre- and in-service training, and more dependent on their ability to master the technological alternatives that refer them to digital sites where relevant, updated subject matter can be accessed. In addition teachers need to be digitally literate facilitators who enhance their students' ability to become autonomous learners and thinkers who possess the digital and computer skills necessary for accessing data relevant to the subject matter they are studying. Furthermore teachers need to develop psycho-pedagogical strategies that can motivate their students to acquire the necessary technological capabilities that will allow them to accurately and efficiently become autonomous learners with the analytic and problem solving ability to access as well as master relevant subject matter in all areas of the school curriculum (following Katz 2014).

3.5 A new informatics curriculum for all students in Poland

The new Informatics Curriculum, introduced in 2015, benefits from our experience in teaching Informatics in schools in Poland for almost 30 years. The first curriculum of

Informatics as an elective subject in high schools was approved by the Ministry of Education in 1985 and this subject has never been removed from the high school curricula. It happened 20 years after the first regular classes on Informatics were held in two high schools in Wrocław and in Warsaw in 1965. Today, Informatics is a compulsory subject in middle (7–9 grades) and in high (10–12 grades) schools and it will replace computer lessons (mainly on ICT) in elementary schools (1–6 grades). The new Informatics Curriculum is also addressed to vocational high schools.

The new unified Informatics Curriculum is addressed to ALL students in K-12 and its main goal is to motivate students to use computational thinking and to engage in solving problems in various school subjects. Moreover its aim is also to encourage and prepare students from early school years to consider computing and related fields as disciplines of their future study and professional career. To this end, the curriculum allows teachers and schools to personalize learning and teaching according to students' interests, abilities, and needs.

The new Computer Science Curriculum (see Sysło and Kwiatkowska 2015) begins with an introduction explaining the importance of Computer Science for our society in general and for our school students in particular. Then follow the curricula for each of the four levels of education (primary 1–3 and 4–6, middle 7–9, and high 10–12). Each curriculum consists of three parts: (I) the first part is a description of purpose of study, formulated adequately to the school level; (II) the second part, which is the same in all curricula levels, includes unified aims which define five knowledge areas in the form of general requirements; (III) the third part consists of detailed attainment targets. The targets grouped according to their aims define the content of each aim adequately to the school level. Thus learning objectives are defined that identify the specific Computer Science concepts and skills students should learn and achieve in a spiral fashion through the four levels of education.

The unified aims are as follows: (1) understanding and analysis of problems based on logical and abstract thinking, algorithmic thinking, and information representations; (2) programming and problem solving by using computers and other digital devices – designing algorithms and programs; organizing, searching and sharing information; using computer applications; (3) using computers, digital devices, and computer networks – principles of functioning of computers, digital devices, and computer networks; performing calculations and executing programs; (4) developing social competences – communication and cooperation, in particular in virtual environments; project based learning; taking various roles in group projects. (5) observing law and security principles and regulations – respecting privacy of personal information, intellectual property, data security, netiquette, and social norms; positive and negative impact of technology on culture, social life and security.

Pedagogically we use a spiral approach based on three elements:

1. problem situations, cooperative games, and puzzles that use concrete meaningful objects – discovering concepts
2. computational thinking about the objects and concepts – algorithms, solutions
3. programming

The emphasis on each of these three elements changes as the students progress.

In the implementation, we suggest three forms of students' activities: visual learning (pictures, objects, abstract and physical models, etc.); auditory learning (exchange ideas, discussions, group work, etc.) and kinaesthetic learning (physical activities).

We are aware of several challenges and the most crucial for the success of the new curriculum are: (1) how to motivate and engage students to develop their computing competences through K − 12, for 12 years (e.g. learning programming requires constant practice); (2) how not to miss a moment when a student can and should decide about her/his future education and career in computing disciplines; (3) what should be the role of programming (we remember that programming is not equal to computing) – when and how to switch from visual to textual programming (visual – for beginners, non-professional, textual – for those who seriously think about Computer Science, since we do not want to lose them)? We see the answers to (1) and (2) in personalization which is aimed at increasing students' interests in learning rather than in studying Computer Science as a discipline, or at least in better understanding how computers and their tools work and can be used in solving problems which may occur in various disciplines. Although any curriculum defines the aims and targets to be included in any school syllabus, the curricula for particular school levels in the new curriculum contain some optional attainment targets which can be freely added to a subject syllabus or assigned only to a group of students. This is a novelty in our national curriculum and leaves some room for personalized learning of gifted students as well as students who have particular interests in specific areas of Computer Science and its applications (such as mathematics, science, arts).

Preparation standards for Informatics teachers at each school level, teacher evaluation criteria and certificates, teaching and learning materials for students and for teachers will accompany the new curriculum. Moreover, all topics in the curricula for other school subject, appropriate for including and using Informatics concepts and skills, have been annotated with comments indicating how to apply computational thinking to enhance knowledge and skills in the other subjects.

4 Emerging themes

Major themes that emerged from these brief vignettes are summarised in Table 3. The themes also echo some of the key ideas that emerged in earlier discussions within IFIP (See Table 1).

Other themes, not included in Table 3, that emerged particularly from the panel discussion but also in some of the vignettes include: reasons and drivers for curriculum change, assessment as a constraint on curriculum development, the need for a more flexible curriculum incorporating personalisation and informal learning, how to incorporate careers advice and the importance of pedagogical considerations in determining curriculum structure and curriculum change. All the emerging themes and some of their interrelationships will be discussed in relation to key ideas from general curriculum theory, mentioned earlier, as they might relate to considerations for the design of the computing curriculum.

4.1 Emerging themes: Entitlement

According to Young (Young 2013), in order to harness the emancipatory capacities of learners, the curriculum should take them beyond their own experience. Thus the goal

Table 3 Emerging Themes in relation to curriculum (in 2015)

Theme	UK	NZ	Australia	Israel	Poland
Entitlement- who is the COMPUTER SCIENCE curriculum for?	All from elementary school upwards	High school subject for seniors; review underway considering requiring it for all students, possibly starting at 5 years old	New curriculum for all	Very talented students BUT *all* must learn computing and technology literacy incorporating computational thinking	All from elementary school upwards
Starting age for COMPUTER SCIENCE	Early	High School (under review as above)	Early	High School	Early
Content balance	Across the content of Computer Science, IT, digital literacy and computational thinking	Under review	Across the content of Computer Science, IT, digital literacy and computational thinking	Computer literacy for all. Computer Science for Some	Across the content of Computer Science, IT, digital literacy and computational thinking
Teacher professional development	A major current challenge	A major current challenge	A major current challenge	Important	A major current challenge

of the curriculum becomes to define its content in a world in which the entitlement to knowledge is the goal. In this endeavour "powerful knowledge" is key, defined as specialised discipline-based knowledge which is different from the experience-based knowledge that pupils bring to school (Young 2013). Clearly, as Young argues, this knowledge is not fixed nor is it equally easily identifiable across all subjects but in Computer Science, as in other disciplines, there are people committed to creating and evaluating the knowledge base, some of which is relatively stable and other aspects are changing quite rapidly.

Answers to the question of entitlement varied across the countries considered in the vignettes above. The Royal Society report emphasised entitlement and opportunity for all students including the "opportunity to learn concepts and principles from Computing" (page 10). Likewise Poland emphasises the importance of opportunity and the goal of motivating all students to use computational thinking and to engage in solving problems as well as to prepare students to consider computing and related fields as disciplines of their future study and professional career. Similarly, the Australian curriculum "Digital Technologies" rationale (http://www.australiancurriculum.edu.au/technologies/introduction) emphasises entitlement as well as economic needs and includes the importance of all students being able to "develop knowledge, understanding and skills to respond creatively to current and future needs". While the UK, Poland and Australia and have taken the firm view that Computer Science within the curriculum is for all, Israel has opted for a segregated model based on students' capabilities. It is noteworthy however that in Israel basic "computing and technology literacy" incorporating computational thinking is an entitlement for all based on the importance of developing problem-solving and analytical thinking skills. This "literacy" goes beyond the typical digital literacy that focuses only on *using* computers rather than problem-solving and computational thinking. New Zealand is actively considering incorporating Computer Science for all from age 5. Young's argument regarding entitlement, outlined above, is that the curriculum question "what knowledge?" is primarily an epistemological one about what should constitute students' entitlement to knowledge (Young 2013).

Learners' entitlement implies entitlement for all and therefore we need to consider: do *all* students need to understand the powerful knowledge in Computer Science that we have begun to identify? There are three particularly compelling arguments, related to entitlement, for Computer Science in compulsory education. First if learners are never introduced to Computer Science as a disciplinary area and to the knowledgebase and approaches that computing academics and professionals use, then they are unlikely to be able to determine whether this is for them, and this is widely regarded as a key factor in the lack of gender diversity in the computing industry (Fisher and Margolis 2003). This therefore is an entitlement and equity issue. Second, as many in the profession have argued, programming is difficult and it takes many years to learn to program (see Robins et al. 2003 for a review). While programming is only one element of Computer Science, it is an essential element and it is inconceivable that an introductory course in Computer Science would not contain programming. Furthermore, while computing professionals do not necessarily do the programming themselves, they need to understand essentials of programming. The third argument is based on the ubiquitous nature of computing: since so much of life today is dependent upon computing everyone needs to develop understanding of Computer Science as well as any computing skills essential for participation in society.

Identification of the powerful knowledge that will constitute entitlement (Young 2013) is evident in the vignettes and the reports mentioned earlier in relation to the development in the UK, Europe and the USA in particular. The Royal Society's description of the discipline of Computer Science encompasses foundational principles, widely applicable ideas and concepts as well as techniques and methods for solving problems and advancing knowledge as well as a distinct way of thinking and working (The Royal Society 2012). Using these headings, key areas of knowledge identified by recent curriculum reports are compared in Table 4. Similarities can be seen in the overview of the discipline, concepts and techniques and methods. Methods and techniques also incorporate ways of thinking and approaching problems. In addition the Joint Informatics Europe & ACM Europe Working Group include the importance of various more general intellectual practices such as tolerance for ambiguity and the ability to communicate and work with others (Joint Informatics Europe and ACM Europe Working Group on Informatics Education 2013). Communication and collaboration are also strands in the Australian and Polish curricula for Computer Science.

Thus we are seeing consensus emerging about the key concepts and techniques of the discipline although perhaps not yet agreement about the importance of more general intellectual practices and social competences. The decision about whether to incorporate these more general skills and practices into curriculum frameworks for Computer Science is a dilemma that is associated with: 1) identifying what is unique about Computer Science and 2) how Computer Science is perceived in relation to other curriculum areas (Denning 2009; Voogt et al. 2015; Bell 2014). Computer Science as a relatively young discipline has suffered from loss of identity owing to various incomplete or inadequate perceptions of its nature (Denning 2009). For example problems have been identified where perceptions of Computer Science become limited to programming or to computational thinking (ibid.).

4.2 Emerging themes: Starting age, order and structure

Following the determination of powerful knowledge, a further step towards defining a curriculum is to determine some order and structure in the content. The age at which students should start to study in this field was an important emerging theme from the vignettes and a key question in determining order and sequence is when do the key knowledge elements need to be introduced? According to Winch (2013) curriculum design is about the management of growth of expertise within a subject. Winch argued that gaining a coherent view of this "epistemic ascent" within a subject by identifying schemata that are at least conceptually and normatively sustainable even if they are not yet empirically ratified is a key element in curriculum design. Moreover Winch argued the need to explore the constraints that the conceptual structure of the subject might impose on pedagogically and cognitively coherent schemata of epistemic ascent and then explore the implications of such constraints within conceptualisations of the subject.

The constraints identified by Winch depend on three interrelated issues. First, it is necessary early in a curriculum to introduce all three major types of knowledge: concepts, propositions and know-how. This is because knowledge of individual propositions implies some understanding of the concepts that such propositions express and this in turn implies a significant ability to understand and make inferences within the

Table 4 Analysis of key areas of knowledge computer science in curriculum reports

	Computer Science Teachers Association (CSTA) Curriculum standards (USA)	Royal Society Report (UK)	Joint Informatics Europe & ACM Europe Working Group (Europe-wide affiliation and focus)
Overview of 'discipline' of Computer Science/ Informatics and foundational principles	The study of computers and algorithmic processes, including their principles, their hardware and software designs, their applications, and their impact on society." (Page 6).	Encompasses foundational principles, widely applicable ideas and concepts as well as techniques and methods for solving problems and advancing knowledge as well as a distinct way of thinking and working (Page 19).	Informatics is a distinct science, characterized by its own concepts, methods, body of knowledge and open issues (Page 3).
Key areas of conceptual knowledge	Strands are: computational thinking; collaboration; computing practice; computers and communication devices; and community, global, and ethical impacts.	Programs, algorithms, data structures, architecture and communication.	Provides the following as examples from a long list of concepts: algorithm; performance and complexity; data structure; concurrency (parallelism) and distribution; language (including programming languages); abstraction.
Techniques and methods	A wide range including techniques and methods for: analysing massive data, solving problems, authentication, recursion, object-oriented, modelling, search, testing.	Computer Science 'methods' or ways of thinking, include: • Modelling: representing chosen aspects of a real-world situation in a computer. • Decomposing problems into sub-problems, and decomposing data into its components. • Generalising particular cases of algorithm or data into a more general-purpose, re-useable version. • Designing, writing, testing, explaining, and debugging programs. (Page 20)	Problem-solving techniques include: • Representing information through abstractions. • Logically structuring and analyzing data. • Automating solutions through algorithmic thinking. • Identifying, analyzing and implementing possible solutions with goal of achieving the most efficient solution. • Formulating problems in a way that facilitates the use of a computer. • Generalizing. (Page 12/13)
Ways of thinking and working	Algorithmic thinking computational thinking scientific thinking logical thinking critical thinking.	computational thinking logical thinking systematic thinking analytical thinking.	Intellectual practices include: • Confidence in dealing with complexity. • Persistence in working with difficult problems. • Tolerance for ambiguity. • Ability to deal with open-ended problems.

Table 4 (continued)

	Computer Science Teachers Association (CSTA) Curriculum standards (USA)	Royal Society Report (UK)	Joint Informatics Europe & ACM Europe Working Group (Europe-wide affiliation and focus)
			• Ability to deal with a mix of both human and technical aspects. • Ability to communicate and work with others to achieve a common goal or solution. (Page 13).
Reference	(Seehorn et al. 2011)	(The Royal Society 2012)	(the Joint Informatics Europe and ACM Europe Working Group on Informatics Education 2013)

subject. This latter knowledge type is *knowing how* to do something. It would not be appropriate, for example, to focus only on practical tasks with the intention of developing know-how without developing understanding of concepts. As we have seen from an analysis of recent curriculum reports (see Table 4) there is general agreement about key concepts and techniques of the discipline. Furthermore it is common for topics in computing curricula to be classified into broad areas that are generally some variation of: programming/algorithms; data representation; digital infrastructure; digital applications and human factors (Duncan and Bell 2015). This reflects the nature of computing: a digital device runs a program (i.e. embodies algorithms); the memory and processor store and manipulates data; components are connected through buses and networks that place limitations on the system; people use them through software and this overall "system" has an impact on individuals and society through issues such as privacy, safety, access to information, and empowering people. For example, grappling with the broad issue of artificial intelligence potentially destroying human society (Future of Life Institute 2015) requires engaging with all of these areas: understanding what the nature of a computer program is (does it "think"?), what kind of knowledge base (data) it could collect and store, its access to our digital infrastructure, the way that humans would interact with it, and the philosophical concerns about human rights and what intelligence is. Many other topics around our digital future require this broad view: government surveillance, secure commerce, the environmental impact of technology (positive and negative), etc.

The second issue according to Winch (2013) is a need for a structured approach to progression in learning the basic facts and central concepts of the subject because knowledge is systematic in terms of 1) classification of its various conceptual elements; 2) the relationships between the elements and 3) the procedures required to gain and validate knowledge. The relationships outlined above between the elements in the computer systems and the human systems within which they operate point towards the need for introducing all of these elements and their relationships as early as possible in order that students may come to develop a broad view. Such an approach might be the spiral curriculum approach explained in the vignette from Poland which aims to develop understanding of all these areas and interrelationships but with different emphases in each stage of education. The arrangement of the specific concepts within a spiral curriculum depends on their complexity as well as the context in which they are introduced and is an area where further empirical research is needed.

Winch's third issue is that the kind of knowledge required to expand and manage subject matter requires a profound understanding of the subject including all of the interacting knowledge types. This therefore is not accessible to school students but comes in more advanced studies beyond school. A constraint following from this third issue requires that the relationship between the ways in which pupils learn by simulating procedures for the acquisition of knowledge in their learning and the actual processes of expansion of disciplinary knowledge should be clarified. For example, project work in computing often involves the systems development life cycle. Winch argues that simulating such procedures may be pedagogically important in developing acquaintance with the knowledge set of the subject as well as building understanding of techniques used in knowledge management. However these simulations should not be seen as simplified versions of expert practice as that might propagate an illusion that high-level design and planning activities are generic and can be used free of the reality

of the skills and materials that are needed to execute the plan. Instead it should be recognised that such expertise requires extensive knowledge and is therefore only possible in higher level courses that build upon previous structured development.

As mentioned earlier there is a view among Computer Science educators that learning programming is difficult but at the same time there is a view emphasised in the New Zealand experience that coming to programming late in students' development is disadvantageous and that if they were to learn some of the techniques, approaches and thinking involved in programming at an earlier stage more students would be successful (Duncan et al. 2014). This therefore is both an entitlement issue for individuals looking to a fulfilling, creative and potentially lucrative career as well as of concern to countries in terms of their economic performance and prosperity. In New Zealand the reintroduction of Computer Science as a high school subject for seniors has highlighted problems associated with this late introduction. Decisions about the early introduction of Computer Science in the UK and Poland were partly based on a view from school teachers and computer scientists in higher education that aspects of Computer Science, especially programming, require gradual acquisition and development over many years. Furthermore there is evidence that exposing students to key computational thinking concepts before 12 years old is not only possible but important for developing their self-efficacy, lack of which can disadvantage girls particularly and thus impact gender diversity in University courses and the IT industry (Duncan et al. 2014).

The evidence that students will be able to learn programming before 12 years old includes: the greater ability to learn natural languages prior to adolescence combined with the similarities between learning natural languages and programming languages; the availability of programming environments and other software designed to support younger learners in learning programming and the experience and views of practitioners who have taught these topics in primary schools (ibid.). Primary school level also affords easier opportunities to apply computing concepts in an integrated environment, whereas subjects tend to be delineated at high school. For example, programming environments for primary school children often use turtle graphics, which also teach coordinate systems, geometry, number concepts as well as ideas around planning and movement, binary representation teaches patterns and properties of whole numbers and leads on to physiological questions such as human perception of colour and sound, and the topic of data compression investigates patterns in human languages.

In summary there is an emerging consensus in Computer Science education that starting to learn Computer Science in primary education, probably around 5 years old, and certainly well before age 12 is not only possible but beneficial for learning as well as developing self esteem and motivation. Furthermore all three types of knowledge need to be introduced early in the curriculum i.e. concepts, propositions and know-how and this can be achieved by a balance of theory and practice. Moreover, given the nature of Computer Science i.e. analysing, designing and developing systems and understanding their human contexts, a spiral curriculum addressing increasingly complex systems and/or introducing more complex concepts is indicated as a curriculum structure.

4.3 Emerging themes: Curriculum content and balance

As outlined previously, consensus has emerged about the key concepts and techniques of the discipline of Computer Science and this translates into curricula that incorporate

broad areas of: programming/algorithms, data representation, digital infrastructure, digital applications, and human factors (Australian Curriculum Assessment and Reporting Authority (ACARA) 2014; Duncan and Bell 2015; Seehorn et al. 2011; The Royal Society 2012). More broadly still there is the issue of balance across Computer Science, IT, digital literacies and computational thinking. As we have seen the UK, Australia and Poland have incorporated elements of all these in their curricula for all students although the balance is only likely to be clear from more detailed analysis of curriculum documents including schemes of work and the curriculum in practice. Previous research comparing Computer Science curricula in different countries revealed the complexity and range of factors affecting the curriculum and how it is implemented and in particular the importance of contextual factors that have resulted in much diversity (Hubwieser et al. 2015).

Another issue emerging from analysis of the five vignettes and associated discussions concerned the importance of more general intellectual practices and social competences. The Polish curriculum incorporates a whole area on developing social competences including project based learning and taking various roles in group projects. The importance of cooperation, collaboration and communication were discussed during the development of the UK curriculum but were not included in the specified content owing to concerns about assessment. IFIP discussions generally agreed that, whereas these intellectual practices and social competences are very important in learning Computer Science, the challenges for their assessment can lead to their under-emphasis. This problem is not only evident in learning Computer Science but more generally in curricula (Webb and Gibson 2015).

4.4 Emerging themes: Teacher professional development

All of the vignettes emphasise the importance of appropriately trained teachers and the major challenge that this provides in countries currently engaged in curriculum change, e.g. UK, Australia, where there are not enough teachers with appropriate knowledge and expertise. The challenge for these countries is threefold. Firstly existing teachers who have taught a very different curriculum may not have sufficient technical knowledge and skills. Secondly their pedagogical content knowledge (Shulman 1987) has not developed in relation to the new curriculum content. Thirdly there are few new Computer Science graduates coming into teaching owing to the general shortage of Computer Science graduates. It is beyond the scope of this paper to examine approaches to teacher professional development in any depth but it is clear that there are resource implications and a need for further research and evaluation of professional development programmes. Early indications, from the vignettes discussed here, suggest that involving teachers in analysing new curricula, creating resources and exploring pedagogical approaches while being supported to develop their own subject and pedagogical knowledge and programming skills can be successful.

4.5 Emerging themes: Risks and drivers of curriculum change

As explained earlier various constraints have been considered in relation to decisions about curriculum change. Change with digital technologies in education is part of a wider global arena that includes bureaucratic as well as commercial and political drivers

(Davis and Mackey 2015). For example, a lack of appropriately qualified and trained teachers increases the risk that the curriculum innovation will fail and therefore inhibits change. In the UK and Australia the discussions emphasised the risk of failure owing to inadequate teaching knowledge and expertise but chose to take the risk and attempt to mitigate it rather than delaying curriculum change. For example, developments in Australia stalled for some months following concerns raised by agencies including teacher unions. Thus decisions about desired curriculum content need to be accompanied by pragmatic considerations including capacity building and economic feasibility. Related to these decisions are considerations of the drivers for curriculum change. In the UK concerns over lack of appropriately trained teachers were high but strong drives from industry prompted political action to drive forward change on the basis of future economic advantages.

Assessment systems can impose serious constraints on curriculum development in the view of many who attended the panel. Assessment is often viewed as existing in a 3-way relationship with curriculum and pedagogy. Thus, where assessment systems impose strong requirements on how assessments may be performed, some curriculum elements may be constrained because they do not lend themselves to the assessment techniques allowed. In this way the creative aspects of Computer Science, for example, may become lost from a curriculum because they are not easily assessed through traditional exams and this concern was expressed by many who attended the panel. Opportunities provided by new computer-based assessments may help to solve this problem but significant research and development challenges remain (see for example Webb and Gibson 2015).

A further set of constraints, which merits research is about how the conceptual structure of the subject might impose on pedagogically and cognitively coherent schemata of epistemic ascent (Winch 2013). We have identified that in different countries there are different views about how well students might cope with concepts and approaches in computing. There is a body of research on learning aspects of computing extending back over at least 30 years but there is a need for continual updating as technological developments not only promote curriculum change but also provide new pedagogical opportunities making some concepts and skills easier to learn. For example new programming languages that simplify the syntax or use visual and/or block-based programming techniques are removing issues that were previously regarded as constraints by making programming easier to learn (Ben-Ari 2013; Mannila et al. 2006; Weintrop and Wilensky 2015). Furthermore there is evidence that students are able to progress from these simpler languages to the more sophisticated languages with more complex syntax generally used in business and industry (Mannila et al. 2006). However the need for teachers to understand not only the programming principles and concepts but also the ways in which these are supported through different languages remains (Ben-Ari 2013).

A further consideration discussed by the panel in Lithuania is how to take account in the school curriculum of increasing out of school and online opportunities for students to develop their computing expertise. Initiatives in some countries have emphasised these in order to make up for perceived deficits in the formal curriculum (Hubwieser et al. 2015). However our view of entitlement discussed above indicated that it is necessary to identify and specify the formal curriculum with an emphasis on entitlement and then extend learning opportunities through technology-supported

environments including school learning platforms, open online courses, community-based maker spaces etc. These online resources can enable "any time" and "anywhere" learning and support teacher-directed activities such as homework, project work, after school clubs etc. as well as providing additional informal learning opportunities for enthusiastic students.

5 Conclusion and recommendations

The brief review of curriculum developments discussed in this paper has identified important areas of consensus. Furthermore it has highlighted issues contributing to the complexity as well as key tensions and constraints in relation to designing a computing curriculum. In summary, there is an emerging consensus about the "powerful knowledge" of Computer Science which is a key element in entitlement (Young 2013). This includes the key concepts of the discipline of Computer Science and of the techniques and methods. There is not yet clear agreement about whether more general elements such as intellectual practices like tolerance for ambiguity and social competences should be specified in a Computer Science curriculum. Differences remain over whether *all* students should learn the discipline of Computer Science. Whereas three of the vignettes presented here and some other examples do emphasise Computer Science for all, the vignette from Israel exemplifies an alternative view in which all students develop basic computing and technology literacy incorporating computational thinking and only very capable students go on to study Computer Science as a discipline. This difference seems to rest on both the difficulties for students in learning Computer Science as well as the identification of the elements of Computer Science that are truly necessary for all students to learn.

In this paper we have standardised the terminology in order to facilitate discussion, there are differences in the terms used in different countries (see Webb et al. 2015 for a comparison of terminology across new developments in curricula).

Constraints that are affecting decisions about the introduction of new curricula include: concerns about students' capabilities and availability of teachers who have appropriate knowledge and skills. As we have discussed, many of the concerns about students' capabilities are being overcome by new pedagogical approaches supported by technological developments. Further research is recommended to identify the relative value of different approaches and their appropriate places in developmental trajectories. These new approaches include new types of programming languages, "unplugged" computing activities as well as more general pedagogical approaches made possible by online opportunities. At the same time we need to continue to investigate how students learn to think computationally and how approaches developed in one curriculum area or context may become generalised to other areas.

Teacher capability has been recognised in this debate as a key limiting factor for curriculum change. Teachers with sound subject and pedagogical knowledge are essential for quality learning. Professional development initiatives for Computer Science Education in some countries aim to develop teachers' subject knowledge alongside their pedagogical knowledge in order to respond to rapid curriculum change. This is unusual; in other subject areas teachers' subject knowledge has typically been developed over a number of years. Thus approaches to professional development for

Computer Science Education as well as new pedagogical approaches are an important area for ongoing research.

Acknowledgments Key presenters at the Curriculum Focus Groups in WCCE 2013 are listed below.
Monique Grandbastien, France
Alain Senteni, UAE and Quebec
Mary Webb, England
Nicholas Reynolds, Australia
Andrej Brodnik, Slovenia
Ivan Kalas, Slovak Republic
Bruria Haberman, Israel
Harriet Taylor, USA

IFIP TC3 Task Force: Curriculum - deeper understanding of roles of Computer Science/ Informatics.
Task Force membership:
Mary Webb, UK (Convenor)
Torsten Brinda, Germany
Andrej Brodnik, Slovenia
Niki Davis, Aotearoa New Zealand.
Andrew Fluck, Australia
Hans Frederik, Netherlands
Monique Grandbastien, France
Pieter Hogenbirk, Netherlands
Ivan Kalas, Slovak Republic
Yaacov J Katz, Israel
Johannes Magenheim, Germany
Peter Micheuz, Austria
Johan van Niekerk, South Africa
Nicholas Reynolds, Australia.
Sigrid Schubert, Germany
Maciej M. Sysło, Poland
Sindre Røsvik, Norway

References

Alsafran, E., & Brown, D. S. (2012). The relationship between classroom computer technology and students' academic achievement. *Research in Higher Education Journal, 15*, 1–19.

Australian Curriculum Assessment and Reporting Authority (ACARA) (2014). Australian Curriculum: Digital Technologies. http://www.australiancurriculum.edu.au/technologies/digital-technologies/Curriculum/F-10?layout=1.

Barak, M., Waks, S., & Doppelt, Y. (2000). Majoring in technology studies at high school and fostering learning. *Learning Environments Research, 3*(2), 135–158.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: what is involved and what is the role of the computer science education community? *ACM Inroads, 2*(1), 48–54.

Bell, T. (2014). Establishing a nationwide COMPUTER SCIENCE curriculum in New Zealand high schools. *Communications of the ACM, 57*(2), 28–30.

Bell, T., & Newton, H. (2013). Unplugging computer science. In D. M. Kadijevich, C. Angeli, & C. Schulte (Eds.), *Improving computer science education* (pp. 66–81). New York: Routledge.

Bell, T., Andreae, P., & Robins, A. (2012). Computer science in NZ high schools: The first year of the new standards. In D. R. M. L. A. S. King, T. Camp, & P. Tymann (Eds.), *43rd ACM technical symposium on computer science education, Raleigh, NC, USA* (pp. 343–348). New York: ACM.

Ben-Ari, M. (2013). Visualisation of programming. In D. M. Kadijevich, C. Angeli, & C. Schulte (Eds.), *Improving computer science education* (pp. 52–65). New York: Routledge.

Cheema, J. R., & Zhang, B. (2013). Quantity and quality of computer use and academic achievement: evidence from a large-scale international test program. *International Journal of Education and Development using Information and Communication Technology, 9*(2), 95–106.

Davis, N. E., & Mackey, J. (2015). *Co-evolutionary perspectives on innovation with digital technologies in education.* Paper presented at the IFIP TC3 Working Conference "A New Culture of Learning: Computing and next Generations", Vilnius, Lithuania, 1–3 July.

Denning, P. J. (2007). The profession of IT: computing is a natural science. *Communications of the ACM, 50*(7), 13–18.

Denning, P. J. (2009). The profession of IT: beyond computational thinking. *Communications of the ACM - One Laptop Per Child: Vision vs. Reality, 52*(6), 28–30. doi:10.1145/1516046.1516054.

Denning, P. J. (2013). The science in computer science. *Communications of the ACM, 56*(5), 35–38. doi:10.1145/2447976.2447988.

Department for Education (2013). *National curriculum in England: computing programmes of study*. London: England.

Duncan, C., & Bell, T. (2015). *A Pilot Computer Science and Programming Course for Primary School students*. Paper presented at the WIPSCE 2015: The 10th workshop in primary and secondary computing education, London, United Kingdom, November 9–11, 2015.

Duncan, C., Bell, T., & Tanimoto, S. (2014). Should your 8-year-old learn coding? In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education* (*WIPSCE 2014*), *Berlin*, *Germany*, (pp. 60–69). 2670774: New York, ACM. doi:10.1145/2670757.2670774.

Fisher, A., & Margolis, J. (2003). *Unlocking the clubhouse: Women in computing*. Cambridge: MIT press.

Future of Life Institute (2015). *Research Priorities for Robust and Beneficial Artificial Intelligence*: *an Open Letter*.

Hubwieser, P., Armoni, M., & Giannakos, M. N. (2015). How to implement rigorous computer science education in K-12 schools? Some answers and many questions. *ACM Transactions on Computing Education, 15*(2), 1–12. doi:10.1145/2729983.

Joint Informatics Europe & ACM Europe Working Group on Informatics Education (2013). Informatics education: Europe cannot afford to miss the boat: Report of the joint Informatics Europe & ACM Europe Working Group on Informatics Education.

Katz, Y. J. (2014). The historical relationship between affective variables and ICT based learning and instruction and achievement in the Israeli school system. In A. T. W. Davey (Ed.), *Reflections on the history of computers in education: Early use of computers and teaching about computing in schools* (pp. 324–338). Heidelberg: Springer-Verlag.

Mannila, L., Peltomäki, M., & Salakoski, T. (2006). What about a simple language? Analyzing the difficulties in learning to program. *Computer Science Education, 16*(3), 211–227. doi:10.1080/08993400600912384.

Naim, Y. (2010). *Attitudes of elementary school pupils and teachers towards a technology-based learning environment*. Ramat-Gan: Bar-Ilan University Unpublished.

Pacheco, J. A. (2012). Curriculum studies: what is the field today? *Journal of the American Association for the Advancement of Curriculum Studies, 8*, 1–18.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books, Inc.

Papert, S. (1996). An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning, 1*(1), 95–123. doi:10.1007/bf00191473.

Reynolds, N., & Chambers, D. P. (2014). Technologies numériques: l'interpretation d' on nouveau curriculum par les enseignants australiens. Revue International D'Education Sevres, 67 Pédagogie et révolution numérique, 63–74.

Reynolds, N., & Chambers, D. P. (2015). Digital Technologies: A new curriculum implementation. In *Society for Information Technology & Teacher Education International Conference*, *Las Vegas*, *March 2015* (pp. 2541–2549).

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: a review and discussion. *Computer Science Education, 13*(2), 137–172. doi:10.1076/csed.13.2.137.14200.

Seehorn, D., Carey, S., Fuschetto, B., Lee, I., Moix, D., O'Grady-Cunniff, D., et al. (2011). *CSTA K–12 computer science standards*. New York: ACM/CSTA.

Shulman, L. (1987). Knowledge and teaching: foundations of the new reform. *Harvard Educational Review, 57*(1), 1–22.

Sysło, M. M., & Kwiatkowska, A. B. (2015). Introducing a new computer science curriculum for all school levels in Poland. In A. Brodnik & J. Vahrenhold (Eds.), *ISSEP, Ljubljana, Slovenia, 2015* (pp. 141–154) . Verlag: Springer.LNCS 9378

The Royal Society (2012). *Shut down or restart? The way forward for computing in UK schools*. London: The Royal Society.

Thompson, D., Bell, T., Andreae, P., & Robins, A. (2013). *The role of teachers in implementing curriculum changes.* Paper presented at the proceeding of the 44th ACM technical symposium on computer science education, Denver: USA.

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: towards an agenda for research and practice. *Education and Information Technologies*, 1–14, doi:10.1007/s10639-015-9412-6.

Webb, M. E. (2014). Considerations for the design of computing curricula. In T. Brinda, N. Reynolds, R. Romeike, & A. Schwill (Eds.), *KEYCIT 2014 – Key Competencies in Informatics and ICT*, *University of Potsdam*, *Germany*, *July 1st – 4th*, *2014* (pp. 267–283): Commentarii informaticae didacticae (CID) | 7.

Webb, M. E., & Reynolds, N. (2013). Panel session: towards a vision for a new curriculum for ICT: are we getting there? at the World Conference on Computers in Education (WCCE), Toruń, Poland, 2–5 July 2013.

Webb, M. E., & Gibson, D. (2015). Technology enhanced assessment in complex collaborative settings. *Education and Information Technologies, 20*(4), 675–695. doi:10.1007/s10639-015-9413-5.

Webb, M. E., Fluck, A., Cox, M., Angeli-Valanides, C., Malyn-Smith, J., Voogt, J., et al. (2015). Thematic working group 9: curriculum - advancing understanding of the roles of computer science/informatics in the curriculum. In K.-W. Lai (Ed.), *EDUsummIT 2015 summary report: Technology advance quality learning for all* (pp. 60–69). Thailand: Bangkok.

Weintrop, D., & Wilensky, U. (2015). *To block or not to block*, *that is the question*: *students*' *perceptions of blocks-based programming.* Paper presented at the proceedings of the 14th international conference on interaction design and children, Boston, Massachusetts.

Wilson, C., Sudol, L. A., Stephenson, C., & Stehlik, M. (2010). Running on empty: the failure to teach K-12 computer science in the digital age. Association for Computing Machinery (ACM), Computer Science Teachers Association (CSTA).

Winch, C. (2013). Curriculum design and epistemic ascent. *Journal of Philosophy of Education, 47*(1), 128–146. doi:10.1111/1467-9752.12006.

Wing, J. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–36.

Young, M. (2013). Overcoming the crisis in curriculum theory: a knowledgebased approach. *Journal of Curriculum Studies, 45*(2), 101–118.